# An Ultra Fast Image Generator (UFIG) for wide-field astronomy

Joel Bergé*, Lukas Gamper, Alexandre Réfrégier and Adam Amara

*ETH Zurich, Department of Physics, Wolfgang Pauli Strasse 27, 8093 Zurich, Switzerland*

**Abstract**

Simulated wide-field images are becoming an important part of observational astronomy, either to prepare for new surveys or to test measurement methods. In order to efficiently explore vast parameter spaces, the computational speed of simulation codes is a central requirement to their implementation. We introduce the Ultra Fast Image Generator (UFIG) which aims to bring wide-field imaging simulations to the current limits of computational capabilities. We achieve this goal through: (1) models of galaxies, stars and observational conditions, which, while simple, capture the key features necessary for realistic simulations, and (2) state-of-the-art computational and implementation optimizations. We present the performances of UFIG and show that it is faster than existing public simulation codes by several orders of magnitude. It allows us to produce images more quickly than SEXTRACTOR needs to analyze them. For instance, it can simulate a typical 0.25 deg$^2$ Subaru SuprimeCam image (10k×8k pixels) with a 5-$\sigma$ limiting magnitude of R = 26 in 30 seconds on a laptop, yielding an average simulation time for a galaxy of 30$\mu$s. This code is complementary to end-to-end simulation codes and can be used as a fast, central component of observational methods relying on simulations.

*Keywords:* Simulations, Wide-field imaging, Computational speed

## 1. Introduction

Image simulations are becoming ubiquitous in observational astronomy. They are intensively used in topics as diverse as extragalactic astrophysics (with public codes like SKYMAKER [4], SIMAGE [13], SHERA [23], or the simulation package developed by the Large Synoptic Survey Telescope (LSST) team [1]), CMB analysis (e.g. [12]), or Supernovae (e.g. [3, 33]). Simulations are mainly used to forecast the results of an observation strategy and to test measurement methods. Examples are given by the LSST simulations [1, 10, 11], the public Shear Testing Program (STEP) and Gravitational Lensing Accuracy Testing (GREAT) simulations [19, 24, 8, 21], as well as individual works (e.g. [2, 20]).

In this work, we focus on simulations used for extragalactic wide-field astronomy. Depending on their aim, such simulations will either be minimalist, like the GREAT simulations which simulated simplistic individual galaxies on a mesh, or include most of, or all relevant cosmology, astrophysics, atmosphere physics and telescope characterization (LSST). The STEP project (using SKYMAKER and SIMAGE) took an intermediate approach, where simulations look realistic, but without focusing much on any particular physics or observational apparatus.

Depending on their emphasis, the speed of simulation codes can vary greatly, from hours to days to simulate a quarter square degree ground-based image. However, if a simulation package is to be used as an integrated part of a method and pipeline calibration process, its speed becomes a driving parameter: a fast simulation code, used to calibrate an external measurement method, allows one to efficiently explore a larger parameter space for the measurement method or survey.

The aim of this paper is to introduce the Ultra Fast Image Generator (UFIG), a fast simulation C++ code able to simulate realistic images on a timescale comparable to that needed by SEXTRACTOR [5] to analyze similar images (i.e., less than one minute for a 0.25 sq. deg. image); since it is widely used in astronomy and is well optimized, we take SEXTRACTOR as a reference. To this end, we adopt simple, yet realistic, models of galaxies, stars and observation conditions, that allow us

*jberge@phys.ethz.ch

to minimize the computation load. We also bring our code to the current computation limits by highly optimizing our implementation through an efficient use of random number generators, parallelization and vectorization. This fast code is thus complementary to end-to-end simulation packages aimed at detailed modeling of observational effects.

Although the code may be used to simulate an image from various ground-based facilities and observation conditions, in this paper we use as a practical test case the simulation of a typical Subaru SuprimeCam [31] coadded 10k×8k pixels image, processed from four 450-seconds exposures, with a 5-$\sigma$ (extended) limiting magnitude of Rc = 26, unless otherwise stated.

Section 2 summarizes the model that we use for galaxies, stars and noise. Further details about it can be found in the appendices. Section 3 motivates the strategy used to optimize the simulation of realistic images, and Section 4 describes the implementation and the optimizations we use on the computational part of the problem; in particular, we present how we optimize random numbers generation and implement multithreading. Section 5 explores the consistency of the UFig simulations with real images. Section 6 shows the performances of our code; in particular, we show in this section how the execution time depends on the image's size and on the exposure time. We conclude in Sect. 7. Further details about UFig, including examples and information about the distribution of the code, can be found at http://www.astro.ethz.ch/refregier/research/Software/ufig.

## 2. Model

The simulations are based on a simple, yet realistic, modeling of galaxies, stars and noise: the models are summarized in this section. The appendices expand on the astrophysics and the methods used to assess our models.

### 2.1. Galaxies

We assume that galaxy profiles are well described by the Sersic profile [38]

$$I(r) = I(r_{50}) \exp\left(-k\left[\left(\frac{r}{r_{50}}\right)^{1/n} - 1\right]\right), \qquad (1)$$

where $r_{50}$ is the radius of the circle enclosing 50% of the total flux of the galaxy, $n$ is the Sersic index, and $k$ is a constant satisfying the equation $2\gamma(2n, k) = \Gamma(2n)$, where $\gamma(x, k)$ is the lower incomplete gamma-function, and $\Gamma(x)$ is the gamma-function.

We find the probability density function (p.d.f) of Sersic indices for bright galaxies (magnitude less than 20) to be well represented by $f(n) = \exp(\mathcal{N}(0.3, 0.5) + \mathcal{N}(1.6, 0.4)) + 0.24$, where $\mathcal{N}(\mu, \sigma^2)$ represents the normal distribution of mean $\mu$ and variance $\sigma^2$ (see Appendix A). For faint galaxies (magnitude bigger than 20), we describe it by $f(n) = \exp(\mathcal{N}(0.2, 1)) + 0.2$.

We parametrize the magnitude distribution of galaxies with a polynomial of the form $\log_{10}(N < \text{mag}) = \sum_i a_i \text{mag}^i$. Table A.2 summarizes the coefficients $a_i$ for different filters.

Finally, we account for the galaxies intrinsic ellipticity distribution with a 2D Gaussian (of both components of the ellipticity) of width $\sigma_1 = \sigma_2 = 0.15$.

### 2.2. Stars and Point Spread Function

We use a Moffat profile to account for the Point Spread Function (PSF). The (circular) Moffat profile is defined as [32]:

$$I(r) = I_0 \left[1 + \left(\frac{r}{\alpha}\right)^2\right]^{-\beta}, \qquad (2)$$

where $I_0$ is the value at the origin ($r = 0$), and $\alpha$ and $\beta$ are scale parameters depending on the observation's conditions. The width of the profile, $\alpha$, is related to its FWHM and to its half-light radius $r_{50}$.

### 2.3. Noise

We finally account for noise in the image. We first add Poisson noise for galaxies and stars. We then add the noise from various sources, like the sky brightness, the readout noise and the errors arising during the data processing, such as flat-field inaccuracies. Following e.g. [17, 29], we define it as a Gaussian random deviate with zero mean and standard deviation given by Eq. (C.1). We finally correlate the noise with a Lanczos resampling [15].

## 3. Code requirements

### 3.1. Requirements

The main requirement of our simulation pipeline is that it must be fast, while providing realistic images. Since SExtractor has become the reference software for wide-field image analysis and is computationally efficient, we use its execution time on a given image as our unit of time. In that sense, we want the running time of the cycle simulation creation (UFig) – simulation analysis (SExtractor) not to be dominated by the execution of UFig, hence setting the requirement that

UFig is not slower than SExtractor. For instance, UFig should simulate a typical Subaru SuprimeCam image of 0.25 sq. deg. (made up of approximately 10,000×8,000 pixels) with a 5-$\sigma$ limiting magnitude of Rc = 26 under one minute.

To meet this goal, we must define the most efficient way to draw galaxies and stars, their generation being the most expensive task of a simulation code.

### *3.2. Pixel-based or photon-based?*

We can think of two ways to simulate 2D objects such as galaxies and stars in an image from a given light distribution: (1) pixel-based and (2) photon-based.

In the former case, an analytic description of the object is pixelized. The description can be a simple profile, which is simply pixelized by taking its value at the center of pixels, or by integrating it in pixels, or it can be more complex, as in a shapelets model [34, 25]. Public simulation packages like Skymaker, SImage or Shera rely on this principle.

In the second approach, an analytical description of the object is considered as the distribution of the photons that make it up. Photons are then drawn individually to make up the object. This approach is used e.g. by the LSST Simulation group [1].

The choice of the algorithm is determined by the total number of operations needed to create all the galaxies and all the stars of the simulation. These are described below.

#### *3.2.1. Number of operations for one elementary building block*

*Pixel-based approach:*. In this case, the elementary building block of an object is one pixel. To simulate one pixel, we first have to draw its value from the analytical description of the object. The value of the PSF for that pixel is drawn in the same way, from an analytical description of the PSF shape. An object should be made on a refined grid (i.e., using pixels smaller than those of the final simulations) in order to minimize approximations of the profile at the center of pixels; analytically integrating over pixels allows one to get rid of those approximations, but at the price of a more complex implementation. Then, Poisson noise must be applied to the pixel. Finally, the object, once created, is convolved with the PSF. This last step, albeit optimized by using FFTs, is computationally expensive, even more if the object is refined so that numerical errors are minimized.

*Photon-based approach:*. In this case, the elementary building block of an object is one photon. To simulate one photon, we draw its position from the analytical description of the object, seen as a distribution function. The effect of the PSF is simply to displace the photon; therefore, we just have to draw a random displacement from the analytical description of the PSF, and apply it to the position of the photon. Contrary to the pixel-based approach, no complex task (such as a convolution) has to be done at all. Finally, since photons are drawn individually, Poisson noise emerges naturally, with no need to add it eventually.

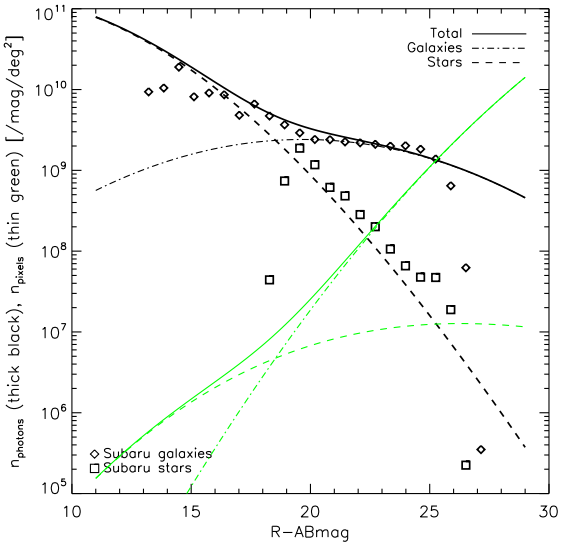


Figure 1: Number of photons (thick black) and number of pixels (thin green) sampled per magnitude per square degree. Dash-dot lines represent the contribution of galaxies, dashed lines show that of stars, and solid lines show the total. Diamonds and squares are photon counts from galaxies and stars from a typical Subaru-SuprimeCam image in Rc-band.

Therefore, less operations are necessary to simulate a photon than to simulate a pixel. In the next subsection, we consider the number of photons and the number of pixels that we need to simulate to make an extragalactic Subaru SuprimeCam-like image, before concluding on what approach we choose.

#### *3.2.2. Number of photons vs number of pixels*

The number of photons coming from galaxies per square degree and per magnitude can be computed from the magnitude distribution of galaxies (Eq. A.7) and the relation between the number of photons and the magnitude of a galaxy (Eq. A.8). A similar approach allows us to estimate the number of photons coming from stars. Integrating those functions, it is easy to estimate the number of photons required for a photon-by-photon simulation. Similarly, assuming that an average galaxy

is contained in a 21×21 postage stamp, and that we resample it by a factor of 5 when simulating it[1], we can estimate the number of sampled pixels per square degree and per magnitude in a pixel-by-pixel approach. The same exercise can be done for stars, assuming that the average postage stamp size is 61×61 pixels (stars are on average brighter and more spread out than galaxies). Fig. 1 shows the number of photons per square degree and per magnitude (black thick lines) and the number of (resampled by a factor 5) pixels required per square degree and per magnitude (green thin lines). In both cases, dash lines correspond to stars, dash-dot lines correspond to galaxies, and solid lines show the total number of photons and pixels. The symbols show photon counts from a typical Subaru image: squares correspond to stars and diamonds to galaxies. Note that the star-galaxy separation, performed with SExtractor, confuses stars and galaxies at magnitudes less than 18, corresponding to saturated objects. Nevertheless, the total number of photons per square degree and magnitude agrees extremely well with our expectations (black lines).

The number of photons per square degree and per magnitude from galaxies remains fairly constant with magnitude, meaning that adding faint galaxies does not affect the execution time when simulating galaxies in a photon-based approach. On the other hand, photons from stars largely dominate the total number of photons at low magnitudes, and become highly subdominant for intermediate and high magnitudes. Thus, in a photon-based approach, the execution time will be dominated by stars generation. The same comparison between stars and galaxies can be done for the number of pixels: stars dominate at low magnitude, before becoming largely subdominant. Contrary to the number of photons, the number of pixels from galaxies per square degree and per magnitude increases linearly with magnitude, meaning that going deeper significantly affects the execution time. In the pixel-by-pixel approach, the execution time is dominated by galaxies.

Table 1 gives the number of photons and pixels that one needs to sample to simulate a 0.25 deg$^2$ image with an 450 seconds exposure time, from magnitude 12 to magnitude 29. These numbers confirm the conclusions from Fig. 1: the number of photons is highly dominated by stars' photons, and the number of pixels is dominated by galaxies. Furthermore, the number of photons needed to simulate galaxies is of the same order than the needed number of pixels.

[1]We find that a factor 5 refinement is a bare minimum, and as such, is a lower bound on what can be used. The bigger this factor, the more precise and expensive the simulation.

Table 1: Number of photons and pixels sampled for one UFig simulation.

| | Number of photons | Number of pixels |
|---|---|---|
| Galaxies | $7\times10^9$ | $5.6\times10^9$ |
| Stars | $5.1\times10^{10}$ | $3.4\times10^8$ |
| Total | $5.8\times10^{10}$ | $6\times10^9$ |

*3.2.3. Conclusion*

The photon-based approach requires less operations per elementary building block (photons), than needed by the pixel-based approach. Furthermore, many more photons than pixels are needed to simulate stars. On the other hand, the numbers of photons or pixels to be sampled in either approach to simulate galaxies are similar. Note that according to Fig. 1, we would have to simulate less pixels than photons, were we to simulate galaxies up to magnitude Rc $\approx$ 26 (corresponding to the Subaru telescope limiting magnitude); however, we need to take into account fainter galaxies, which affect the image's noise, and we choose Rc = 29 as our higher magnitude.

For these reasons, we have decided to adopt a hybrid approach: we simulate galaxies with a photon-based approach, and stars with a pixel-based approach.

## 4. Implementation

### *4.1. Galaxies*

We simulate a circular Sersic galaxy by sampling the radial position $r$ of its $N_\Phi$ photons (Eq. A.8 links a galaxy's magnitude and its number of photons) from a $\gamma$-distributed random variable $Y$ ($r = Y^n r_{50}/k^n$), and their angular positions from a uniform distribution between 0 and $2\pi$ (see Appendix A). Then, we transform the galaxy so that it becomes elliptical with the desired ellipticity, as shown in the appendix.

Finally, the galaxy is pixelized by truncating the coordinates of the photons' positions. Note that we truncate, and not round, the coordinates because a pixel can be seen as a bucket (e.g., $x = 2.8$ corresponds therefore to pixel number 2).

### *4.2. PSF-induced photon displacement and stars*

The effect of the PSF on a incident photon is to displace it, the corresponding displacement following a p.d.f defined by the PSF profile. This can be seen by considering stars as point-like sources, which would appear as a Dirac function in the absence of a PSF; their

observed shape (the PSF itself) is therefore the consequence of the displacement of the photons making up the oncoming Dirac function. The displacement d$X$ due to the PSF is then obtained by uniformally sampling the Moffat profile's cumulative distribution function (c.d.f) $\mathrm{d}X = \alpha\sqrt{[cdf(Y)-1]^{\frac{1}{1-\beta}}-1}$ (see Appendix B). We use this technique to convolve galaxies with the PSF.

As shown in Sect. 3, stars account for most of the photons to be simulated, hence we create them with a pixel-based approach. To this end, we integrate the Moffat profile numerically in pixels with a 7th order Legendre-Gauss quadrature rule [27], and we perturb each pixel's value with a Poisson deviate. We integrate the profile from the center outwards, until the probability of detecting one photon in a pixel is less than one percent. While much faster, this method is statistically equivalent to drawing photons one by one. Furthermore, as opposed to a pixel-based approach to galaxy generation, creating stars pixel by pixel does not involve expensive numerical convolution with the PSF, nor is it impacted by potential numerical errors coming from the convolution.

### *4.3. Optimizations*

#### *4.3.1. Random number generation*

Operations such as drawing the position of galaxies and stars, drawing galaxies' photons, or generating the background noise, imply heavy use of random number generators.

To enable the creation of the $\approx 3 \times 10^{10}$ random numbers needed to simulate galaxies[2], we implemented the lagged Fibonacci generator $x_i = (x_{i-21034} + x_{i-44497}) \bmod 2^{32}$ [6, 7], that we initialize with the mersenne twister generator implemented in the boost::mt19937 random number generator [26]. The lagged Fibonacci generator is buffered, meaning that we generate a full lag of 44497 numbers at once instead of generating them only when needed. Finally, another advantage of the lagged Fibonacci generator is that it allows us to use vectorization (see below) to generate the lag. We tested our generator with the Test U01 test suite, which consists of 160 tests [22]: all tests were passed with a p-value inside the range $[10^{-4}, 1-10^{-4}]$; tests with a p-value outside of this range would be considered as failures.

[2]This number corresponds to four times the number of photons that must be generated (see Table 1), since a photon requires four random numbers: position with respect to the galaxy's center (radial and angular) and displacement due to the PSF (radial and angular).

#### *4.3.2. Parallelization*

We start by generating a catalog of stars and galaxies, where astrometry, photometry and shape information is stored. This task is done with an openmp loop.

Then, we implement multithreading in the following way. Galaxies are sorted by position (each thread thus dealing with a well defined region of the full image), in such a way that each thread gets the same number of photons; in this way, all threads run in the same time (a very bright galaxy does not impair the speed of a given thread). Stars are sorted by position, by making sure that each thread gets the same number of stars to generate. Since galaxies and stars are sorted by position, each thread works safely on its own part of the simulation, completely independently from the other threads. Therefore, we do not need critical section (openmp locks) to write to the global array making up the image. Locks are further avoided by forcing each thread to have its own set of random number generators (independent of other threads' random number generators).

The remaining tasks (noise generation, magnitude rescaling, image resampling) are parallelized using openmp.

#### *4.3.3. Other optimizations*

*Approximation of functions.* We use linear interpolations to common functions such as trigonometric functions or $\Gamma$ function. This significantly speeds up our calculations.

*Vectorization.* Streaming SIND Extensions (SSE) allows us to perform four floating point calculations at once: we use it for galaxy generation, noise generation and image resampling. We checked that using floating point values instead of double points value does not impact the precision.

## 5. Quality assurance

Fig. 2 compares a patch of a UFig simulation (lower panel) with a patch of a typical Subaru image (upper panel). Both patches are of the same size, and the dynamic scale is the same in both panels. Visually, the shape and size of galaxies are well rendered by our simulations. Moreover, the granularity and spatial correlation of the background is comparable to that of the real image. We simulate correlated noise by resampling the original simulation, whose background noise is uncorrelated, normally distributed, with a Lanczos resampling. This resampling is fast (see Sect 6), and allows us

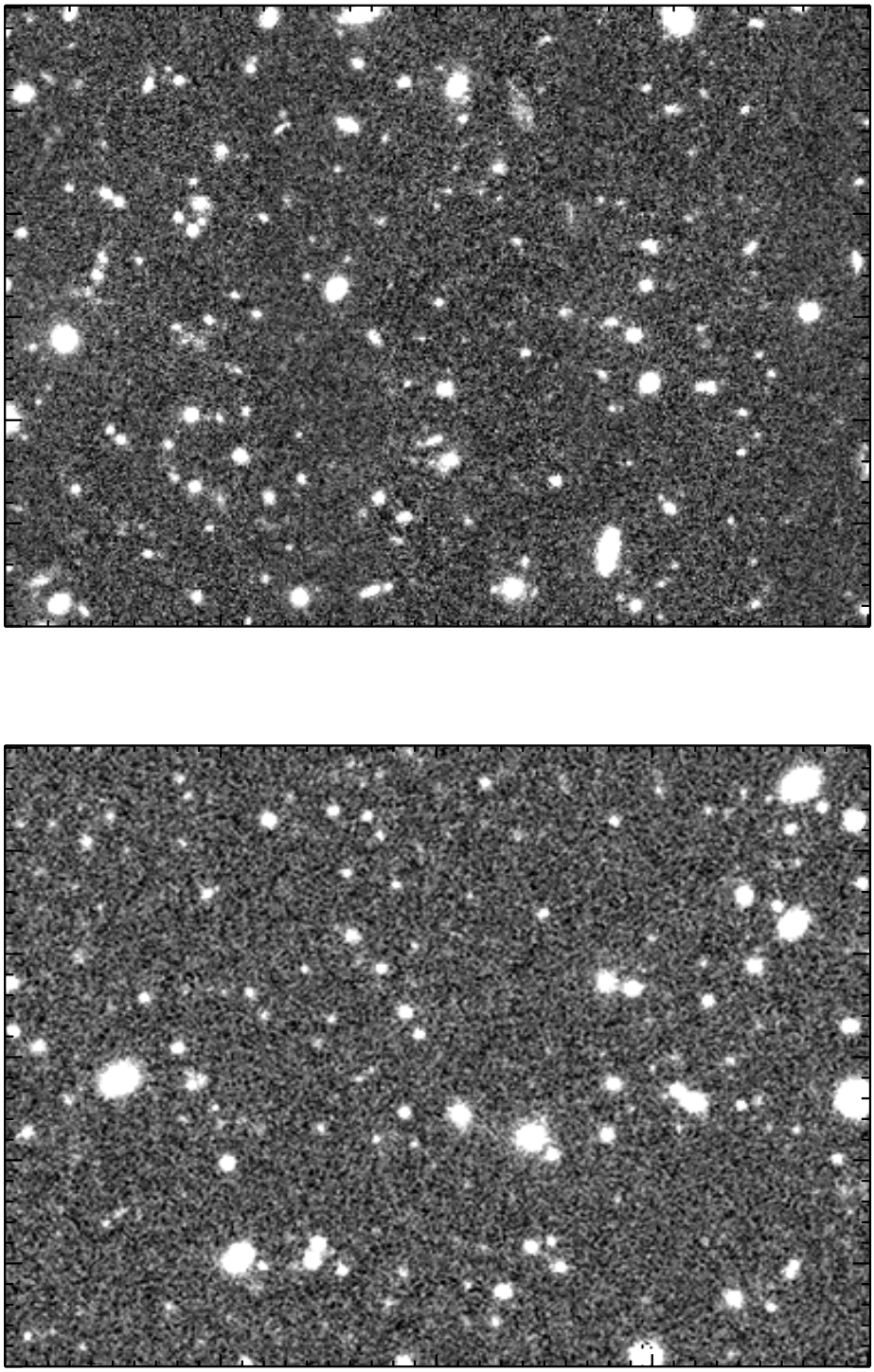

Figure 2: Top: typical Subaru image. Bottom: UFIG simulation. Both patches have an area of 400×300 pixels$^2$, or 1.3×1 arcmin$^2$. The dynamic scales are the same for both panels.

to mimic the resampling done in the processing of real data.

In Fig. 3, we show the distribution of pixels in the real image (black solid line) and in the simulation (red dashed line). The distributions agree well, especially for values near zero, corresponding to background pixels, and for high-value pixels, corresponding to bright objects. Around zero, the distributions are well rendered by Gaussian distributions, hence confirming that generating the background noise from a Gaussian deviate is correct. The flat distribution of negative values for the real image are created when processing raw data single exposures into a resampled, coadded image. Although they are not reproduced with the original Gaussian noise model used in UFig (not shown on the figure), we have checked that they appear when we simulate raw data single exposures that we process to obtain a final coadded image. This confirms that they are indeed due to the data processing. Resampling the (uncorrelated Gaussian noise-) simulated image with a Lanczos resampling allows us to better mimic the data reduction process and to better reproduce the background noise, while avoiding an expensive data reduction of simulated raw images.

Fig. 4 shows the distribution of stars and galaxies in the magnitude-size plane, both for a typical Subaru image (upper panel) and our simulation (lower panel). Magnitudes are given by SExtractor, and we define the half-light radius ee50 as SExtractor's FLUX_RADIUS (note that all SExtractor's parameters are the same when running SExtractor on the real image and on the simulation, preventing any difference from the SExtractor analysis). We also rely on SExtractor to perform the galaxy-star separation; to this end, we set SExtractor's SEEING_FWHM equal to the seeing input in the simulation (0.6" in the case shown by Fig. 4), and we further set SExtractor's CLASS_STAR=0.9. Galaxies are shown by the black symbols, while stars are shown by the green symbols. The star branch is narrower in the simulation than in the real image because we forced the PSF to be constant in the simulation.

Despite the simple models used, UFig produces realistic images, that are consistent with real images.

## 6. Computational performance

### *6.1. Execution time*

We tested the performance of UFig on a laptop (macBook Pro with a 2.7 GHz Intel processor, 4 cores and 16 GB RAM). Using eight threads, UFig provides a 10k×8k-pixels-image (approximately the size of a typical Subaru image) in 30 seconds. This is smaller than

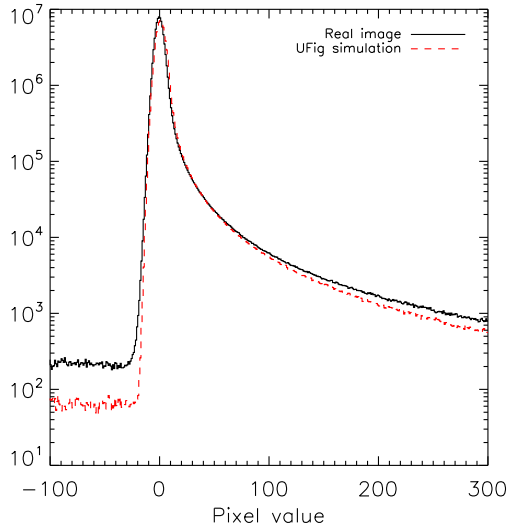


Figure 3: Pixels distribution. The solid black line is for a typical Subaru image, and the dashed red line is for a UFig simulation.

the execution time of SExtractor on a similar image. For comparison, it took several hours to create a similar image using Skymaker and Simage.

We further measured that UFig uses 66% of the laptop's peak performance (29 Gflops out of 43.2 peak Gflops).

In the remainder of this section, we discuss how the UFig execution time depends on some input parameters, and how the time spending is distributed between different tasks. Fig. 5 shows how the execution time to create an image with exposure time $t_{\rm exp}$ = 450s depends on the image's size, when using eight threads. The black thick solid line shows the total execution time. The red thick short dashed line shows the time spent sampling galaxies, while the green thick dash-dot line shows that spent simulating stars. The dash-dot-dot-dot line corresponds to the time needed to generate noise and the long dashed line to the time spent resampling the image. Diamonds represent the time spent writing the image and the corresponding catalog to the disk. Finally, the dotted line shows the overheads (defined as all tasks not directed directly at creating or writing the simulation). For big enough images (more than $10^7$ pixels), most of the time is spent drawing galaxies, then drawing stars. For smaller images, the execution time is dominated by overheads. Excepted the overheads, that are expected not to depend strongly on the image size, all tasks' time-spending show a clearly linear dependence on image

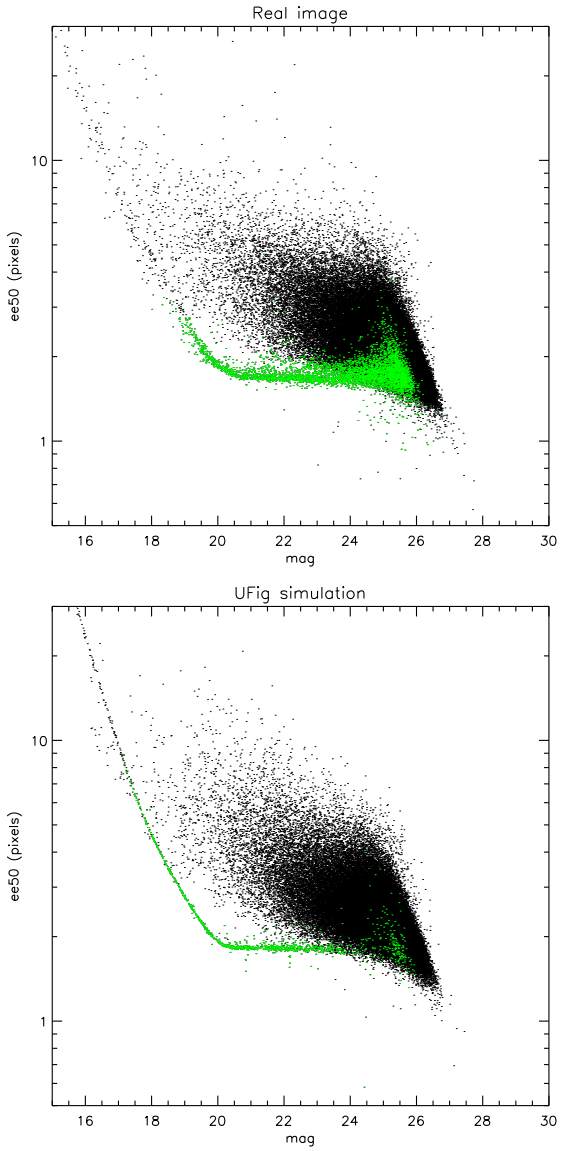


Figure 4: Magnitude-size distribution for a typical Subaru image (upper panel) and for a UFig simulation (lower panel). Black points represent galaxies, and green points represent stars, galaxies and stars being separated by SExtractor.

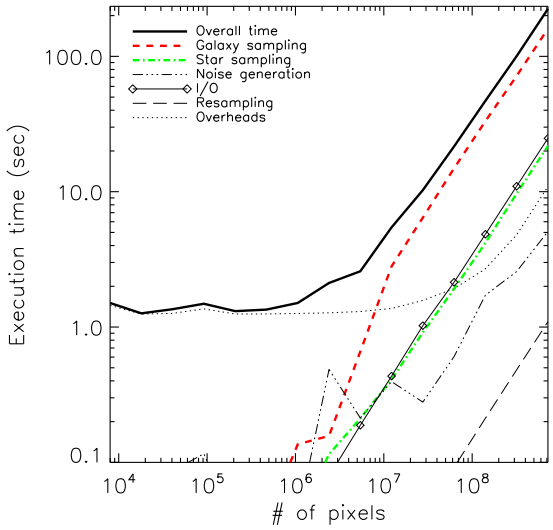


Figure 5: Execution time vs image size, using eight threads. Black solid line: overall time. Red short dashed line: galaxy sampling. Green dash-dot line: stars sampling. Dash-dot-dot-dot line: noise generation. Diamonds: image and catalog writing to disk. Long dashed line: image resampling. Dotted line: overheads

size. This behavior is expected, since the number of photons (for galaxy generation) and pixels belonging to stars (for star generation) increase linearly with the image size. Similarly, all other tasks, by definition, depend linearly on the number of pixels.

Fig. 6 shows the relation between the exposure time used in the simulations, and the execution time, when using eight threads. Lines have the same meaning as in Fig. 5. The time spent resampling the image is not shown, since it is less than one second. For small exposure times (less than 200 seconds), most of the time is spent writing to disk, whereas generating galaxies is most expensive for large exposure times. Generating galaxies and stars both scale linearly with exposure time. This is expected; the number of photons that one has to simulate to generate galaxies obviously depends linearly on the exposure time. So does the flux, and therefore the number of pixels that one has to draw when creating stars. All other tasks do not depend on the exposure time.

### *6.2. Parallelization*

Fig. 7 shows how the execution time depends on the number of threads used for the computation, when simulating a 10,000×8,000 pixels image with an exposure

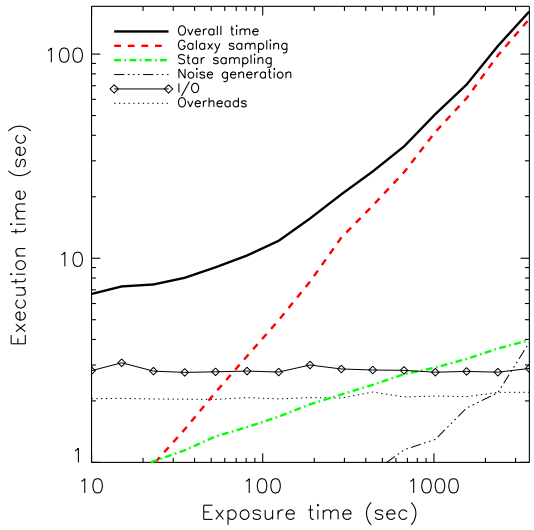


Figure 6: Execution time vs exposure time, using eight threads. Line styles and colors are the same as in Fig. 5.

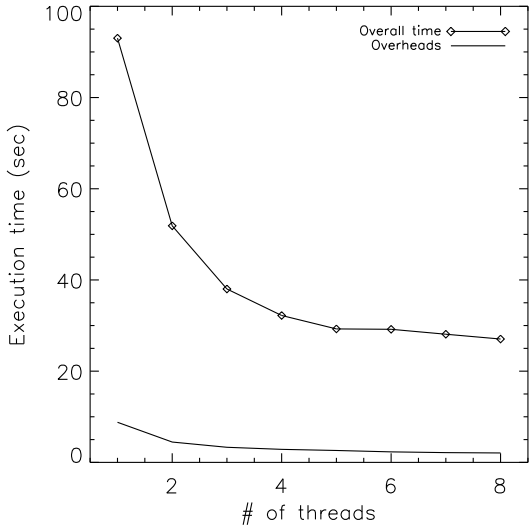


Figure 7: Execution time vs number of threads. Diamond-solid line: overall execution time. Thin solid line: overheads.

time $t_{\rm exp}$ = 450 s on the same laptop as that used above. The lower curve corresponds to the overheads, as defined above. When using two threads, we need half as much time to create a simulation as when using only one thread. A significant gain in execution time can be seen when using up to five threads, before the execution time plateaus. This plateau is due to the fact that Intel hyper-threading is at work, meaning that when more than four cores are used, threads starts to compete against each other (our laptop having four physical cores – corresponding to eight virtual cores). We therefore expect an optimal parallelization to provide us with a factor of four improvement in execution time: this is indeed what we measure, meaning that our parallelization is nearly optimal.

### *6.3. Memory management*

Independently of the number of threads used, a run of the code uses the amount of RAM memory needed to store one copy of the simulated image. For instance, for a 10,000×8,000 pixels simulation, 300 MB of memory are required.

## 7. Conclusion and perspective

By introducing the Ultra Fast Image Generator (UFIG), we showed that it is currently possible to implement very fast codes to simulate wide-field astronomical images. We showed that, using simple models, we can simulate realistic images which take observation constraints, including the PSF and various sources of noise, into account.

Combining analytic simple models with state-of-the-art computing optimizations allows us to produce the mock of a typical Subaru SuprimeCam image (0.25 sq. deg, 10k×8k pixels) with a 5-$\sigma$ (extended) limiting magnitude of Rc = 26 , in which we account for galaxies of magnitude up to Rc = 29, in 30 seconds when using a laptop (macBook Pro with a 2.7 GHz Intel processor, 4 cores and 16 GB RAM); thus, an average galaxy is simulated in 30 $\mu$s. This represents an improvement of several orders of magnitude in execution time compared to the public softwares that we are aware of. It is also comparable with the execution time of SExtractor on a similar image; given the optimization of its implementation, as well as its extensive and intensive use, SExtractor can be taken as a standard, well optimized, code in astronomy, thus setting a timescale for any new software that is used in combination with it. UFIG is thus complementary to end-to-end simulation codes which aim to model observational effects in great details, but with greater execution time.

We have presented the implementation of the code, with an emphasis on the different optimizations that we use. In particular, we have found that the optimal solution is to adopt a hybrid approach to generate galaxies and stars, where we create galaxies with a photon-based approach, and stars with a pixel-based approach. We

have also described how we optimize random number generation by implementing a lagged Fibonacci random number generator, how we parallelize the code using multithreading in which threads are completely independent, and how we approximate common functions and use SSE vectorization to speed up calculations.

We have then shown that UFig's simulations are consistent with real images, using simple standard tests. Finally, we have investigated the performances of the code, where we have checked that the execution time scales as it should were the code perfectly optimized. In particular, we showed that the parallelization of the code is nearly optimal. Therefore, the UFig implementation reaches the limits of the current computation possibilities. This is highlighted by the usage of 66% of our laptop's peak performance by UFig.

The current UFig implementation relies on simple models; although it produces images sufficiently realistic for many applications, such as testing data processing codes, or calibrating photometry or astrometry codes, these simple models may need to be refined to use UFig for very high-precision analyses. Therefore, we plan to increase the UFig realism by including more cosmology in the code. For instance, we will increase the shape complexity of galaxies, allow them to be spatially clustered, as well as distributed in redshift, and we will add weak lensing (either from large-scale structures or from massive clusters). Using more complex models will have an impact on the code's performance, which we will assess and take into account to keep UFig optimal.

Another application of UFig's speed is to calibrate a computer intensive measurement method. For instance, [35] and [20] showed that a promising approach to cosmic shear measurement pipelines is to calibrate them with image simulations (with observation conditions similar to those of the real data to analyze) to alleviate systematic effects. Up to now, such a calibration was time-consuming and thus limited. Hence, UFig opens a new window to improve on computational intensive measurement techniques, such as those used in weak lensing, or in transient searches, for which UFig's ability to efficiently simulate time series observations may prove to be central.

Further details about UFig can be found at http://www.astro.ethz.ch/refregier/research/Software/ufig.

## Appendix A. Galaxy model

### *Appendix A.1. Sersic profile*

We describe galaxies by a Sersic profile (Eq. 1). To assess the distribution of the Sersic index for galaxies up to high redshift, we use the Advanced Camera for Survey General Catalog (ACS-GC – [18]). Figure A.8 shows the distribution that we extract from the catalog, as measured in the I-band, for galaxies with good photometric redshifts and magnitude between 15 and 26. Colors code for different ranges of magnitudes: black for magnitude less than 20, red for magnitude between 20 and 22, green for magnitude between 22 and 24, and blue for magnitude between 24 and 26. For bright galaxies, the distribution is clearly bimodal, as is well known (see e.g. [14]). This bimodality disappears for fainter galaxies. Whether this highlights physical processes in galaxy formation and evolution, or whether it is simply due to a selection effect, is beyond the scope of this paper: we are only interested in modeling the distribution of Sersic indices as close to the observed one as possible. We find that the p.d.f of Sersic indices for bright galaxies (magnitude less than 20) is well represented by:

$$f(n) = \exp(\mathcal{N}(0.3, 0.5) + \mathcal{N}(1.6, 0.4)) + 0.2. \quad \text{(A.1)}$$

For faint galaxies (magnitude bigger than 20), we find that the p.d.f of Sersic indices if well described by:

$$f(n) = \exp(\mathcal{N}(0.2, 1)) + 0.2. \quad \text{(A.2)}$$

This analytical description is shown for faint galaxies as a dashed line in Fig. A.8.

#### *Appendix A.1.1. The circular Sersic profile seen as a $\gamma$-distribution*

The Sersic profile is given by Eq. (1), and can be normalized to unity flux as:

$$I(r) = \frac{k^{2n}}{2\pi n r_{50}^2 \Gamma(2n)} \exp\left[-k\left(\frac{r}{r_{50}}\right)^{1/n}\right]. \quad \text{(A.3)}$$

This profile can be seen as the p.d.f of the radial position of photons from a circular galaxy. Then let $X$ be the random variable describing the position of a photon. The probability to find the photon in the shell $[X, X+\mathrm{d}X]$ from the center of the galaxy is given by

$$f_X(X)\mathrm{d}X = 2\pi X I(X)\mathrm{d}X. \quad \text{(A.4)}$$

Defining the random variable $Y = k\left(\frac{X}{r_{50}}\right)^{1/n}$, it can be shown that it follows a $\gamma$-distribution of shape parameter $2n$:

$$f_Y(Y) = Y^{2n-1}\frac{e^{-Y}}{\Gamma(2n)}. \quad \text{(A.5)}$$

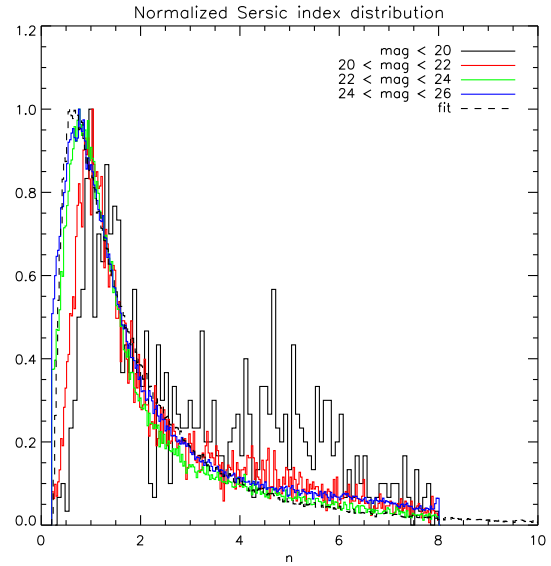


Figure A.8: Distribution of Sersic indices from the ACS-GC catalog [18], for different magnitude ranges. Histograms are scaled such that their maximum is 1. The dashed line shows the analytical description input in UFig for galaxies with magnitude bigger than 20.

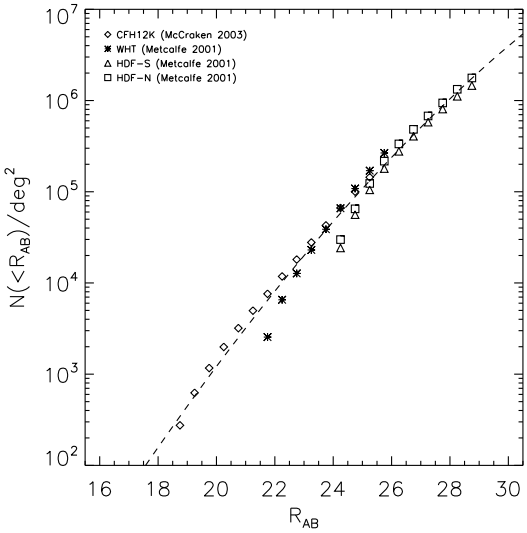


Figure A.9: Cumulative distribution of galaxies' magnitude in the R-band, from several surveys. The fitting function (dashed line) is defined by Eq. (A.7) and Table A.2.

*Appendix A.1.2. From a circular to an elliptical galaxy*

We define a galaxy's ellipticity $(\varepsilon_1, \varepsilon_2)$ through its quadrupoles $J_{ij}$ as $\varepsilon_1 + i\varepsilon_2 = (J_{11} - J_{22} + 2iJ_{12})/(J_{11} + J_{22})$

A circular galaxy can be made elliptical through the transformation $(x_\mathcal{E}, y_\mathcal{E}) = A(x_C, y_C)$, where $(x_\mathcal{E}, y_\mathcal{E})$ and $(x_C, y_C)$ are the photon's coordinates in the elliptical and circular galaxy respectively, and $A$ is the transformation matrix (for $\|\varepsilon\| \neq 0$):

$$A = \frac{1}{\sqrt{2}} \times \begin{pmatrix} \mathrm{sign}(\varepsilon_2)\sqrt{1+\|\varepsilon\|}\sqrt{1+\frac{\varepsilon_1}{\|\varepsilon\|}} & -\sqrt{1-\|\varepsilon\|}\sqrt{1-\frac{\varepsilon_1}{\|\varepsilon\|}} \\ \sqrt{1+\|\varepsilon\|}\sqrt{1-\frac{\varepsilon_1}{\|\varepsilon\|}} & \mathrm{sign}(\varepsilon_2)\sqrt{1-\|\varepsilon\|}\sqrt{1+\frac{\varepsilon_1}{\|\varepsilon\|}} \end{pmatrix} \quad \text{(A.6)}$$

If $\|\varepsilon\| = 0$, $A$ is the identity matrix.

*Appendix A.2. Magnitude distribution*

We parametrize the magnitude distribution of galaxies from galaxy counts in different surveys, such as the VIRMOS Descartes [28], COSMOS [9], SXDS [16], and the Hershell Telescope and Hubble Deep fields [30]. We compile the cumulative counts of these surveys, and fit the resultant overall counts with a polynomial of the form

$$\log_{10}(N < \mathrm{mag}) = \sum_i a_i (\mathrm{mag} - 23)^i, \quad \text{(A.7)}$$

Table A.2: Fitting coefficients for galaxy cumulative counts.

| Filter | $a_0$ | $a_1$ | $a_2$ |
|---|---|---|---|
| Rc | 4.300 | 0.383 | -0.00766 |
| I | 4.579 | 0.360 | -0.0229 |
| Z | 4.558 | 0.410 | -0.0248 |

as shown by Fig. A.9 for counts in the R-band. Table A.2 summarizes the coefficients $a_i$ for different filters.

The number of photons making up a galaxy of magnitude mag, for a single exposure time $t_{\mathrm{exp}}$, in the AB-magnitude system, is given by:

$$N_\Phi = 10^{-26} t_{\mathrm{exp}} \Delta S \frac{\Delta\lambda}{h\lambda} T 10^{0.4(8.9+\beta \mathrm{airmass} - \mathrm{mag})}, \quad \text{(A.8)}$$

where $\Delta S$ is the effective telescope's mirror's surface, $\lambda$ is the filter's central wavelength, $\Delta\lambda$ the filter's width, $\beta$ is the atmospheric extinction, $h$ is the Planck constant, and $T$ is the total throughput of the observation system $T = T_{\mathrm{mirror}} T_{\mathrm{camera}} T_{\mathrm{filter}} T_{\mathrm{corrector}}$.

This number can be rescaled to simulate a galaxy on a coadded image with magnitude zero-point $\mathrm{mag}_0$:

$$N'_\Phi = N_\Phi 10^{0.4(\mathrm{mag}_0 - \mathrm{mag}_0^{(I)} - 2.5\log(t_{\mathrm{exp}}))} \quad \text{(A.9)}$$

where $\mathrm{mag}_0^{(I)}$ is the instrument magnitude zero-point, i.e. the magnitude corresponding to a flux of 1 ADU for a one second exposure time in the same observing

conditions,

$$\mathrm{mag}_0^{(I)} = 8.9 + \beta \mathrm{airmass} - 2.5 \log \frac{\mathrm{gain}}{Q_E 10^{-26} \Delta S \frac{\Delta\lambda}{h\lambda} T}, \tag{A.10}$$

where gain is the CCD's gain and $Q_E$ its quantum efficiency.

*Appendix A.3. Magnitude-size relation*

We use the ACS-GC catalog to parametrize the relation between the apparent magnitude and apparent size of galaxies. In this catalog, galaxies' $r_{50}$ are estimated by fitting a Sersic profile. Since this catalog is shallower than the public data we used to estimate the magnitude distribution (Appendix A.2), we do not use it to parameterize the magnitude distribution, but use that derived above. The ACS-GC data are therefore used only to analytically describe the relation between the magnitude and the size of galaxies. We find that in the $\mathrm{mag}_r - r_{50,r}$ plane, where $\mathrm{mag}_r$ and $r_{50,r}$ are the magnitude and the size rotated such that they become uncorrelated, and shifted such that they have zero mean, the distribution of the log of the size $r_{50,r}$ at a given magnitude $\mathrm{mag}_r$ is well fitted by a Gaussian. Additionally, given that the correlation angle between the size and the magnitude is small, we checked that the magnitude distribution derived in Appendix A.2 for the unrotated magnitudes is still a good fit to that of the rotated magnitudes $\mathrm{mag}_r$.

Thus, we set a galaxy's magnitude and size such that:

$$\begin{pmatrix} \mathrm{mag} \\ r_{50} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \mathrm{mag}_r \\ r_{50,r} \end{pmatrix} + \begin{pmatrix} \mathrm{mag}_p \\ r_{50,p} \end{pmatrix} \tag{A.11}$$

where $\mathrm{mag}_p$ and $r_{50,p}$ set the pivot point around which magnitudes and sizes are rotated. They are estimated from the magnitude and size means of COSMOS, and are set to $\mathrm{mag}_p = 25.309$ and $\log r_{50,p} = -0.796$ arcsec. The correlation angle $\theta$ is set to 5.7 deg. The rotated magnitude $\mathrm{mag}_r$ is drawn from the distribution (A.7), and the log of the rotated size $r_{50,r}$ is drawn from a normal distribution of zero mean and standard deviation 0.19 arcsec.

We checked that the magnitude-size relation does not depend significantly on the observing band, and therefore, we restrict its parametrization to what is presented here.

## Appendix B. Stars and Point Spread Function models

We use a Moffat profile to account for the PSF. The (circular) Moffat profile is defined in Eq. (2), where $I_0$ is the value at the origin ($r = 0$), and $\alpha$ and $\beta$ are scale parameters depending on the observation's conditions. For instance, for realistic atmospheric turbulences, $\beta = 4.765$ [39]. We use $\beta = 2.6$ to account for instrumental effects (especially diffraction). The profile's width $\alpha$ is related to its FWHM by $\alpha = \frac{\mathrm{FWHM}}{2\sqrt{2^{1/\beta}-1}}$ and to its $r_{50}$ by $\alpha = \frac{r_{50}}{\sqrt{2^{1/(\beta-1)}-1}}$.

Contrary to the case of the Sersic profile, the Moffat profile cannot be linked to a usual known distribution, from the p.d.f of which we can easily estimate the displacement to apply to a given photon in UFig. Therefore, we sample the displacement by inverting the c.d.f of the profile.

The Moffat profile, seen as the normalized p.d.f $f_X(X)$, where the random variable $X$ corresponds to the photon's displacement, is given by:

$$f_X(X)\mathrm{d}X = \frac{2(\beta-1)X}{\alpha^2}\left[1 + \left(\frac{X}{\alpha}\right)^2\right]^{-\beta} \mathrm{d}X. \tag{B.1}$$

Defining the variable $Y = 1 + (X/\alpha)^2$, so that the p.d.f of $Y$ is $f_Y(Y) = (\beta-1)Y^{-\beta}$ for $Y > 1$, and integrating it, we obtain the c.d.f of $Y$:

$$cdf(Y) = 1 - Y^{1-\beta}. \tag{B.2}$$

The displacement due to the PSF is then obtained by inverting Eq. (B.2), the c.d.f. itself being uniformally sampled:

$$X = \alpha\sqrt{[cdf(Y) - 1]^{\frac{1}{1-\beta}} - 1}. \tag{B.3}$$

We parametrize the stars' magnitude distribution with a polynomial fit to the Milky Way model derived in [36]. Given the position in the sky of the image we want to simulate, we extract the stars' magnitude distribution from the corresponding online application [37].

## Appendix C. Noise model

Poisson noise is automatically and naturally accounted for when simulating galaxies in a photon-based approach. We add Poisson noise for stars, and account for the noise from various sources, like the sky brightness, the readout noise and data processing, with a Gaussian random deviate with zero mean and standard deviation (see e.g. [17] or [29]):

$$\sigma_N = \sqrt{n_{\mathrm{exp}}\left(\frac{RON}{\mathrm{gain}}\right)^2 + \frac{F_{\mathrm{sky}}}{n_{\mathrm{exp}}\mathrm{gain}} + f_{\mathrm{dp}}}, \tag{C.1}$$

where *RON* is the readout noise of the camera, $F_{\rm sky}$ is the sky brightness in ADUs, $n_{\rm exp}$ is the number of exposures out of which the coadded image is assumed to be done and $f_{\rm dp}$ describes the noise coming from the data reduction (including, but not limited to, flat-fielding inaccuracies).

It should be noted that independently of the hybrid approach we use to simulate galaxies and stars, we treat the background noise at the pixel level, and therefore add it in ADUs to the noiseless image. Finally, we resample our simulations with a Lanczos filter to better mimic the data reduction process.

## Acknowledgements

We want to thank Stefan Müller and Julien Carron for useful discussions, as well as Jason Rhodes and Richard Massey for comments on the manuscript. We acknowledge the usage of the Jet Propulsion Laboratory supercomputers, which are provided by funding from the JPL Office of the Chief Information Officer, and we thank Jason Rhodes for making this usage possible.

## References

[1] LSST Science Collaborations, Abell, P. A., Allison, J., et al. 2009, arXiv:0912.0201
[2] Bergé, J., Price, S., Amara, A., & Rhodes, J. 2012, MNRAS, 419, 2356
[3] Bernstein, J. P., Kessler, R., Kuhlmann, S., et al. 2012, ApJ, 753, 152
[4] Bertin, E. 2009, Memorie della Società Astronomica Italiana, 80, 422
[5] Bertin, E., & Arnouts, S. 1996, A&AS, 117, 393
[6] Brent, R.P., 1992, Proc. of Fifth Australian Supercomputer Conference, Melbourne, Dec. pp. 704-706
[7] Brent, R. P. 1994, Mathematics of Computation, 63, 389
[8] Bridle, S., Balan, S. T., Bethge, M., et al. 2010, MNRAS, 405, 2044
[9] Capak, P., Aussel, H., Ajiki, M., et al. 2007, ApJS, 172, 99
[10] Chang, C., Kahn, S. M., Jernigan, J. G., et al. 2012, arXiv:1206.1378
[11] Chang, C., Marshall, P. J., Jernigan, J. G., et al. 2012, arXiv:1206.1383
[12] Delabrouille, J., Betoule, M., Melin, J.-B., et al. 2012, arXiv:1207.3675
[13] Dobke, B. M., Johnston, D. E., Massey, R., et al. 2010, PASP, 122, 947
[14] Driver, S. P., Allen, P. D., Graham, A. W., et al. 2006, MNRAS, 368, 414
[15] Duchon, C. E. 1979, Journal of Applied Meteorology, 18, 1016
[16] Furusawa, H., Kosugi, G., Akiyama, M., et al. 2008, ApJS, 176, 1
[17] Grazian, A., Fontana, A., De Santis, C., et al. 2004, PASP, 116, 750
[18] Griffith, R. L., Cooper, M. C., Newman, J. A., et al. 2012, arXiv:1203.1651
[19] Heymans, C., Van Waerbeke, L., Bacon, D., et al. 2006, MNRAS, 368, 1323
[20] Kacprzak, T., Zuntz, J., Rowe, B., et al. 2012, arXiv:1203.5049
[21] Kitching, T. D., Balan, S. T., Bridle, S., et al. 2012, MNRAS, 423, 3163
[22] L'Ecuyer, P., Simard, R., 2007 ACM Transactions on Mathematical Software, Volume 33 Issue 4
[23] Mandelbaum, R., Hirata, C. M., Leauthaud, A., Massey, R. J., & Rhodes, J. 2012, MNRAS, 420, 1518
[24] Massey, R., Heymans, C., Bergé, J., et al. 2007, MNRAS, 376, 13
[25] Massey, R., & Refregier, A. 2005, MNRAS, 363, 197
[26] Matsumoto, M., Nishimura, ., 1998, ACM Transactions on Modeling and Computer Simulation: Special Issue on Uniform Random Number Generation, Vol. 8, No. 1, January 1998, pp. 3-30.
[27] http://mathworld.wolfram.com/Legendre-GaussQuadrature.html
[28] McCracken, H. J., Radovich, M., Bertin, E., et al. 2003, A&A, 410, 17
[29] Meneghetti, M., Melchior, P., Grazian, A., et al. 2008, A&A, 482, 403
[30] Metcalfe, N., Shanks, T., Campos, A., McCracken, H. J., & Fong, R. 2001, MNRAS, 323, 795
[31] Miyazaki, S., Komiyama, Y., Sekiguchi, M., et al. 2002, PASJ, 54, 833
[32] Moffat, A. F. J. 1969, AAP, 3, 455
[33] Perrett, K., Balam, D., Sullivan, M., et al. 2010, AJ, 140, 518
[34] Refregier, A. 2003, MNRAS, 338, 35
[35] Refregier, A., Kacprzak, T., Amara, A., Bridle, S., & Rowe, B. 2012, arXiv:1203.5050
[36] Robin, A. C., Reylé, C., Derrière, S., & Picaud, S. 2003, A&A, 409, 523
[37] http://model.obs-besancon.fr/
[38] Sersic, J. L. 1968, Cordoba, Argentina: Observatorio Astronomico, 1968
[39] Trujillo, I., Aguerri, J. A. L., Cepa, J., & Gutiérrez, C. M. 2001, MNRAS, 328, 977